to it, but the subset of the original idea would probably be not patentable anymore, unless there is an unexpected result or synergy somewhere.

[0082] The expiration date is shown by a clock/calendar on screen, which also affects the value of patent, as decreasing versus time. The value of the patent can also be variable depending on the market, e.g. demand and supply ratio, or offers/auctions. Some of the choices are: common, no-open-source, or both-patent-and-open-source. In Admin option, one can change the fundamental, basic, or boundary parameters.

[0083] The parameters for the game can be set before, and stored, with a file name, to be recalled later, for specific game and corresponding pre-set parameters. The unit for scores is dollar, point, or any other real or virtual counting system. The bank account may have negative balance, which can become positive again by adding the funds. Or, one can put a min. limit of zero balance on it.

[0084] The interactive patent system described above with reference to FIGS. **1-8** may be developed with software systems known to one skilled in the art. For example, Ruby on Rails ("Ruby") may be used to develop the interactive patent system of FIGS. **1-8**. Ruby is a web-application and persistence framework that includes everything needed to create database-backed web-applications according to the Model-View-Control pattern of separation (as an example). This pattern splits the view (also called the presentation) into "dumb" templates that are primarily responsible for inserting pre-built data in between HTML tags. The model contains the "smart" domain objects (such as Account, Product, Person, Post) that holds all the business logic and knows how to persist themselves to a database. The controller handles the incoming requests (such as Save New Account, Update Product, Show Post) by manipulating the model and directing data to the view.

[0085] In Ruby, for example, the model is handled by what's called an object-relational mapping layer entitled Active Record. This layer allows you to present the data from database rows as objects and embellish these data objects with business logic methods. The controller and view are handled by the Action Pack, which handles both layers by its two parts: Action View and Action Controller. These two layers are bundled in a single package due to their heavy interdependence. This is unlike the relationship between the Active Record and Action Pack that is much more separate. Each of these packages can be used independently outside of Ruby, as an example.

[0086] By default, Ruby will try to use Mongrel and lighttpd web servers if they are installed. Otherwise, Ruby will use the WEBrick, the webserver that ships with Ruby. When you run script/server, Ruby will check if Mongrel exists, then lighttpd and finally fall back to WEBrick. This ensures that you can always get up and running quickly. Mongrel is a Ruby-based webserver with a C-component (which requires compilation) that is suitable for development and deployment of Ruby applications. If Mongrel is not installed, Ruby will look for lighttpd. It is considerably faster than Mongrel and WEBrick and also suited for production use, but requires additional installation and currently only works well on OS X/Unix, as an example. And finally, if neither Mongrel nor lighttpd are installed, Ruby will use the built-in Ruby web server, WEBrick. WEBrick is a small Ruby web server suitable for development, but not for production.

[0087] It is also possible to run Ruby on any platform that supports FCGI. Apache, LiteSpeed, IIS are just a few. More information on FCGI may be found at http://wiki.rubyonrails.com/rails/pages/FastCGI, as an example.

[0088] The Ruby on Rails code used to create the interactive patent system used in FIGS. **1-8** is included as a computer program listing appendix on CD accompanying this application. This code is meant as one example of possible code that may be used to carry out the present invention; Other systems could be written in other languages or using other frameworks. Each section of code begins with #<filepath>, wherein <filepath> describes the file hierarchy in which the code may exist. If the particular <filepath> is empty, a second entry marked #directory empty" will be present in the code. These comments are meant to permit a user to recreate not only the code, but also the specific file and folder hierarchy used for the example shown in FIGS. **1-8**.

[0089] While the above description describes a graphical, computer-based game, the game may be created to operate as a text-based game as well. Such a text-based game may be useful when a player is available only on, for example, email or text messaging. Similarly, the game could be adapted for use with sound prompts and vocal commands (e.g., for visually impaired users or users without access to a computer screen or the like).

[0090] As noted above, not only could money (or points) be a factor in the interactive patent system of the present invention, but time may also be considered a factor. The exchange currency is another option for more realistic costs and fees for different jurisdictions around the world. Time delays may be incorporated for patent prosecution, freedom to operate determinations, product design, and patent enforcement. These time delays may be particularly useful when the interactive patent system of the present invention is incorporated into a virtual world. Time can also be a factor that measures skill of the user because some users may be able to complete tasks in the patent simulation more quickly than others, thus gaining a competitive advantage in accumulating points (or money, or whatever is used to represent scoring in a particular simulation).

[0091] Here is a description of the administrative interface and its key features. The administrative interface for the patent game allows users to trace and analyze the course of a given game. At the simplest level, it offers users a quick and easy way to view and query the data stored in the database for a set of games. The interface can display, for any given game, the game type, the players involved in the game, what innovations were available to the players, what patents were filed, what licenses were granted, and what critical events took place during the game.

[0092] The events portion of the interface is particularly useful for users trying to follow the course of a game. When put together in chronological order, the collection of events for a given game forms a rough narrative describing the players' interactions. An event might be something like, "Player A patented 'ABC'," or, "Player B licensed 'ABC'." All interactions that affect the state of the game are logged as a single-sentence event in the database, and maybe displayed to all or some users.

[0093] The quantitative information stored in the database, such as the number of patents and licenses created during a game, allows users to perform statistical and numerical ana-